CLASSIFICATION USING NEURAL NETWORKS AND DEEP LEARNING

**Mahitha Boddu**  
Masters in Computer Science  
Arizona State UniversityTempe, Arizona, USA, 85281  
mboddu@asu.edu

# Introduction

The primary objective of this project is to build and train a convolutional Neural Network for the classification of images from the SVHN Dataset. Neural Networks enables computer to learn from observational data, they are set of algorithms which are designed for pattern recognition. Data such as images, sound, text are translated into numerical patterns for the network to recognize. Deep Learning are set of techniques to learn Neural Networks. Convolutional Neural Networks( CNN) are Neural Networks essentially used for classification, clustering of images and object recognitions like people, digits, symbols, etc. CNN’s are regularized versions of multilayer perceptron. In CNN the network learns to optimize the kernels through automated learning and so require little pre-processing compared to other image classification algorithms. It primarily consists of three layers, input layer, hidden layer and the output layer. All the layers other than input and output layers are termed as the hidden layers. The layers present in the architecture of CNN are convolutional layers, pooling layers, fully connected layers. Convolutional layer is the building block of the network, they convolve the input and send to the next layer. They also do the job of dimensionality reduction. Pooling layers combine the outputs at one layer to a single neuron in the next layer there by reducing the dimensions of the data. There are two ways of pooling local and global, local pooling combines the small clusters and global pooling is for all neurons of feature map. Further most commonly used pooling methods are average pooling and max pooling, where the former takes the average value of input image partitions and the latter takes the maximum value in the input partitions. Fully connected layers do the job of classification in the CNN, where this layer connects every neuron in one layer to the next layer. Activation functions used within the layers are used to introduce the nonlinearity into the network. Rectified Linear Unit (ReLU), sigmoid, hyperbolic tangent function are used as activation functions to increase the nonlinearity. Generally, ReLU activation is desirable since it trains the neural network many times faster without remarkable penalty to accuracy. The data used in the project for the classification is the “Street View House Numbers” dataset abbreviated as SVHN dataset. It is a real-world dataset of images for developing object oriented, machine learning algorithms. These are similar to MNIST data and requires minimal formatting, data processing and is taken from house numbers in Google Street View Images. Dataset consists of 73257 training digits, 26032 testing digits, 531131 additional training data and has two formats. One format is the original images with character level bounding boxes and the other being 32x32 images centered around single character.

# Description of the solution

## Project Description

In this project, I have used SVHN Dataset for building and training convolutional Neural Network for the purpose of classification. The SVHN Dataset contains printed digits pertaining to ten classes. Constructed the Convolutional Neural Network with the architecture provided which consists of convolutional layer, max pooling layer, fully connected layer. I used Keras, an open source software library which provides a python interface for the Neural Networks. This Keras library contains various modules and layers like Conv2D, MaxPooling2D, Dense, Flatten and so on which can be used for the construction of CNN. The architecture of CNN used for the project is shown in the figure below.

Graphical user interface, text, chat or text message

Description automatically generated

There are different parameters associated with each layer of the network. Our architecture with the parameters is a combination of convolution and MaxPooling alternatively followed by Dense (fully connected) layers.

First Layer: Convolutional layer with 64 feature maps, kernel size of 5x5 and using ReLU (Rectified Linear Unit) as activation function followed by the second layer.

Second Layer: MaxPooling layer with kernel size of 2x2 and stride 2x2 followed by the next convolutional layer.

Third Layer: Convolutional layer with 64 feature maps, with kernel size of 5x5 and ReLU activation function.

Fourth Layer: MaxPooling layer with 2x2 kernel size and stride followed by the next convolution layer.

Fifth Layer: Convolutional layer with 128 feature maps, with kernel size of 5x5 and ReLU activation function.

Sixth Layer: Next is the fully connected layer with 3072 nodes or units and ReLU activation.

Seventh Layer: Fully connected layer with 2048 units and ReLU activation.

Eighth Layer: The final layer is the fully connected layer with softmax activation function and 10 output nodes.

The main objective of convolutional layer is to extract the high-level features from the input image of size 32x32, we use kernel size of 5x5 for this purpose and it is also used for dimensionality reduction. Kernel moves over the image and extracts the high-level features from the image which are close to the characteristics of the original image and the movement of kernel over the image is shown in the figure below.

Diagram

Description automatically generated

The next layer which is the pooling layer also does the same work as the convolutional layer but on the reduced dimensions. Pooling is of two types, max pooling and average pooling where the max pooling gives the better accuracy. Max Pooling layer chooses the maximum value for each kernel size by striding along with the mentioned stride size. After the feature extraction, before passing these features to the fully connected layer it is sent to the Flatten layer to flatten the input. The output of the convolutional layers is converted to single long feature vector (flattened) before sending it to the fully connected layer. The single column flattened output vector is sent to the fully connected layer where it does the mathematical operation of mapping the data image to non-linear function. The activation function used in our architecture for the fully connected layers is the ReLU activation function which is defined as g(z) = max(0,z). After the fully connected layers with ReLU activation, the last fully connected layer uses softmax activation function since it maps the output to [0,1] in such a way that the total sums up to 1 so the output of softmax is a probability distribution. Hence this softmax activation function is used to find the probabilities of input being a particular class (classification). After the model creation and construction of the layers, the model is then compiled which is the final step in the creation of model. Important arguments for compiling the model are the loss function, optimizer and the metrics. Where the loss function is used to find the deviation or the error in the process of learning, optimizer is used to optimize the input weights by comparing the loss function and prediction, and metrics are used for the performance evaluation of the model. Finally, the accuracy of the testing dataset is calculated.

## Design and Implementation Details

Before proceeding with the algorithm, some parameters used in implementation are fixed as below.

* Batch size = 128
* Epochs = 20
* Kernel size = 5x5 for convolution layers, 2x2 for max pool layers
* Stride = 1x1 for convolution layers, 2x2 for max pool layers
* Stochastic gradient descent optimizer with a learning rate of 0.07

The algorithm followed to complete the classification using convolutional Neural Networks is as follows.

* Imported the required Keras modules and the training, testing datasets. Normalized the data, encoded the labels using one hot vector encoding.
* Sequential model is created using the “Sequential API” of imported Keras library and all the convolution, MaxPool, fully connected layers are added to the model according to the architecture.
* Then the created sequential model is compiled using the loss function as categorical\_crossentropy, optimizer as Stochastic gradient descent and the metrics as accuracy.
* The model is then trained using the fit function, to evaluate the model and from this testing and training accuracies, losses can be found. Classification accuracy, loss values are obtained and plotted the graphs with respect to number of epochs.

# Results

The classification accuracy obtained with the above-mentioned parameters is 91.49% and a test loss of 0.5111. Whereas the training accuracy and loss obtained are 99.93% and 0.0075, respectively. The plots obtained for the accuracy and loss as a function of number of epochs are as follows:

Chart

Description automatically generated with medium confidence

A picture containing graphical user interface

Description automatically generated

Training loss and testing loss are plotted in the same graph as a function of epochs. The output and initial, final loss values for an epoch count of 20 can be summarized as follows:

|  |  |  |
| --- | --- | --- |
|  | Epoch count = 1 | Epoch count = 20 |
| Training loss | 2.0976 | 0.0075 |
| Testing loss | 0.8453 | 0.5111 |

Table

Description automatically generated

# Contribution to the project

This is an individual projects and the entire work is done by me to complete the project. Since, the project is based on the course modules of Neural Networks and Deep learning, I first tried to get the complete understanding of the concepts explained in the module like perceptron model, activation function, back propagation, simple convolutional neural networks, concepts of kernel, stride, regularization. Thorough knowledge of these concepts helped me in understanding the project easily. Since I am new to this domain I have read about the software and libraries required to complete the project. Learned about the Keras, which is an open source, deep learning framework for python and completed few tutorials to understand it’s usage.

Then I used Google Colab for implementing the project, by importing Keras modules necessary for the Neural Network architecture and also importing the SVHN Dataset. Constructed the Convolutional Neural Network with the architecture of convolution, MaxPooling, fully connected layers using conv2D, maxPooling2D, Dense layers of Keras respectively. After constructing the model, I compiled it using Stochastic Gradient Descent optimizer and then trained the model using the fit function to find the classification accuracy and the testing loss for the dataset used. I have made few changes in the model design and tried to check how the graphs and output differ. The results are as follows:

* Adding Batch Normalization after the convolutional and fully connected layers in the model. Adding this the learning becomes efficient and overfitting can be avoided. The result of this is as follows:

A picture containing rectangle

Description automatically generated

Chart, line chart

Description automatically generated

* In the implementation I have used a batch size of 128 but since smaller batch sizes give better results I have reduced the batch size to 32 and noted the results. Found that the test accuracy is a bit higher for batch size of 32.

A picture containing graphical user interface

Description automatically generated

A picture containing graphical user interface

Description automatically generated

Table

Description automatically generated

# Lessons Learned and team Members

There are many algorithms which are used for classification, but Convolutional Neural Networks are best suitable for recognizing sizes, shapes, digits. Though this method requires significant training time, they run faster and requires less space.

* Unlike other algorithms the accuracy and loss/error for this classification algorithm depends on hyperparameters and various other parameters.
* Learning rate is a hyperparameter which manages how much the model needs to be changed when weights of the model are updated in response to the estimated error. Choosing this value of hyperparameter is a challenging task, since a large value for the learning rate results in unstable training process by learning the sub-optimal set of weights and a small value may lead to long training process which may get stuck.
* So, learnt that the “learning rate” hyperparameter is the most important aspect while implementing the neural networks and in our project, Convolutional neural network. It is configurable and the value ranges between 0 and 1.
* The optimization used in our project, Stochastic Gradient Descent estimates the gradient error for the current state and updates the weights of the model using back propagation algorithm.
* Batch size is also one of the important hyperparameters which influences the algorithm and is defined as the number of samples used in the estimate of error gradient from the training dataset.
* As the batch size increases it takes less time to train per epoch, but it is generally recommended to use smaller batch size since they reduce the generalization error, regularization effect. Batch size of 32 is generally preferred.
* Batch Normalization is a regularization technique which is used to overcome the problem of overfitting. It normalizes the input of every layer and allows the layers to learn independently and also stabilizes, accelerates the process of learning.
* The epoch count decides the number of times the weights of the network are changed. As long as training and testing errors decreases with the increase in epoch count, it can be incremented, and it is not one of the most important parameters in learning.
* The argument called decay can be used in the in the Stochastic Gradient Descent optimizer which uses the learning rate. With decay we can model how the learning rate gets updated for each iteration.

# References

1. <http://ufldl.stanford.edu/housenumbers/>
2. <https://wiki.pathmind.com/convolutional-network>
3. <https://www.tutorialspoint.com/keras/index.htm>
4. <http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf>
5. <https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/>
6. <https://colab.research.google.com/notebooks/welcome.ipynb>
7. <https://keras.io/getting_started/>
8. <https://en.wikipedia.org/wiki/Convolutional_neural_network>
9. <https://analyticsindiamag.com/everything-you-should-know-about-dropouts-and-batchnormalization-in-cnn/>